VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

MOBILNÍ HRA S POUŽITÍM ROZŠÍŘENÉ REALITY

MOBILE GAME USING AUGMENTED REALITY

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ADAM KONEČNÝ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. PAVEL NAJMAN**
SUPERVISOR

BRNO 2018

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Konečný Adam**
Obor: Informační technologie
Téma: **Mobilní hra s použitím rozšířené reality**
Mobile Game Using Augmented Reality
Kategorie: Uživatelská rozhraní

Pokyny:

1. Prostudujte principy interakce v rozšířené realitě.
2. Seznamte se s platformou iOS a s možnostmi tvorby her pro tuto platformu.
3. Navrhněte mobilní hru využívající prvky rozšířené reality.
4. Implementujte navrženou hru.
5. Otestujte vytvořenou hru a sesbírejte zpětnou vazbu od uživatelů.

Literatura:

- Dle pokynů vedoucího.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Najman Pavel, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Cílem práce bylo vytvoření hry v rozšířené realitě pro operační systém iOS. Výsledná hra je založena na herním žánru tower defense. Práce se věnuje rozšířené realitě a jejímu vývoji, dále popisuje nástroje potřebné pro vytvoření této hry. Další část práce popisuje návrh a následnou realizaci tohoto projektu. Nakonec se věnuje dostupným nástrojům pro testování, testování samotnému a rozebere výsledek testování.

Abstract

Main goal of this thesis was to create a game in augmented reality running on iOS operating system. The game is based on the tower defense game genre. The thesis describes augmented reality and its history. The work also describes tools which are necessary to create this project. Following part describes the design and development process of the game. Lastly the thesis deals with the tools available for testing, testing itself and its results.

Klíčová slova

Rozšířená, realita, iOS, Unity, Xcode, ARKit, hra, obrana, věž, iPhone, iPad

Keywords

Augmented, reality, iOS, Unity, Xcode, ARKit, game, defense, tower, iPhone, iPad

Citace

KONEČNÝ, Adam. *Mobilní hra s použitím rozšířené reality*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Pavel Najman

Mobilní hra s použitím rozšířené reality

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Pavla Najmana. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Adam Konečný
15. května 2018

Poděkování

Děkuji svému vedoucímu bakalářské práce panu Ing. Pavlu Najmanovi za jeho cenné rady, které mi dal, čas, který mi věnoval a v neposlední řadě taky za jeho trpělivost. Dále bych rád poděkoval svému kamarádovi Tomáši Svozilovi za vytvoření grafiky pro uživatelské rozhraní hry.

Obsah

Kapitola 1

Úvod

Od příchodu chytrých telefonů se firmy snažily tlačit jejich výkon co nejvíce dopředu. Ten se posledních pár let více a více přibližuje výkonu počítačů. V kombinaci s pokročilými senzory těchto zařízení, velikostí a přenositelností se z nich staly ideální nástroje pro rozšířenou realitu.

Vzhledem k tomu, že mám zkušenosti s vývojem aplikací pro platformu iOS a zároveň mě baví dělat hry v herním enginu Unity, rozhodl jsem se tyto tři věci zkombinovat a vytvořit hru v rozšířené realitě běžící na platformě iOS.

Hra je typu *tower defense*, jejíž cílem je to, aby hráč zabránil nepřátelům projít celou mapu tím, že k cestě staví věže, které do nepřátel střílí a snaží se je zabít. Za každé zabití získává odměnu, kterou využije na další stavění a vylepšení věží.

V této práci vysvětluji co to rozšířená realita je, zmíním její historii, aktuální stav, využití v různých odvětvích a také popíši některé aktuálně dostupné aplikace využívající rozšířenou realitu.

Poté vysvětlím jakým způsobem funguje rozšířená realita konkrétně na platformě iOS a jak se vyvíjí aplikace pro iOS obecně. Pozornost je věnována samotnému procesu vydání aplikace na App Store a co vše je k tomuto úkonu potřebné. Dále se věnuji hernímu enginu Unity a implementaci rozšířené reality v této platformě.

Další dvě kapitoly se věnují konkrétnímu návrhu hry, jejímu fungování a popisu co vše v ní je možné. Také je zde popsán celkový návrh kódu. Potom je popsána implementace hry s důrazem na podstatné prvky hry.

V poslední kapitole je velká pozornost věnována průběhu testování hry, nástrojům, které byly použity pro testování a získávání zpětné vazby a v neposlední řadě se věnuje sesbíraným datům z testování.

Kapitola 2

Rozšířená realita

Termínem rozšířená realita se popisuje obraz skutečného světa, který je obohacen o počítačově vytvořené 2D nebo 3D prvky. Ty jsou do světa zasazeny tak, aby vytvářely iluzi, že jsou opravdovou součástí tohoto světa čehož se dosahuje například tím, že vrhají stín na skutečné objekty nebo intenzita a teplota barev světla působícího na tento objekt odpovídá co nejpřesněji skutečnému světlu [3].

2.1 Historie rozšířené reality

Může se zdát, že je rozšířená realita velice mladou technologií, ale není to ve skutečnosti pravda [8]. Už v roce 1968 na Harvadu vyvinul vědec Ivan Sutherland systém nositelný na hlavě (*Head-mounted display*) [12] vytvářející jakousi vrstvu s počítačem generovaným obsahem nad skutečným světem. Obsah se nijak neuzpůsoboval podle natočení hlavy a celkově tyto systémy neměly žádné praktické využití.

1974 - Myron Krueger vytvořil laboratoř nazvanou "Videoplace". Kombinoval projektory s videokamerami, které emitovaly různé obrazce a uživatelé tak byli obklopeni interaktivním prostředím.

1992 - Louis Rosenberg vyvinul "Virtual Fixtures"pro Air Force [7]. Jeden z prvních systémů využívajících rozšířenou realitu. Jednalo se o vnější kostru, ve které člověk vzdáleně ovládal stroje a prováděl nějaké úkoly, zatímco měl na hlavě systém pro rozšířenou realitu, díky které viděl různé překážky. Vnější kostra zajišťovala, aby se člověk nemohl pohnout dál než bylo dovolené a pomocí rozšířené reality viděl kam se ještě pohnout může a kam už ne.

1999 - Hirokazu Kato vytvořil open-source knihovnu umožňující rozpoznávání specifických značek v obraze a jejich pohyb a natočení v čase. Nad obraz se vykreslovaly virtuální objekty jejichž velikost, pozice a rotace odpovídaly značce v obraze. Obdobného principu bylo využito u reklamy BMW popsnané níže.

V roce 2008 vyšla aplikace *Wikitude AR Travel Guide* pro mobilní telefony využívající kameru zařízení. Na displeji byl obraz z kamery rozšížený o relevantní informace k okolní krajině. Uměla zobrazit třeba názvy hor nebo názvy zámků a k nim krátký popis. Také uživatele informovala o jeho vzdálenosti od daného místa.

Téhož roku využila automobilka BMW rozšířenou realitu pro reklamní účely. V časopisech se objevila reklama se speciálním obrázkem, kterou když uživatel držel před počítačovou kamerou, viděl na obrazovce auto umístěné na této reklamě. Speciální software v obraze tento obrázek hledal a pokud ho našel, umístil na něj 3D model auta. Toto umož-

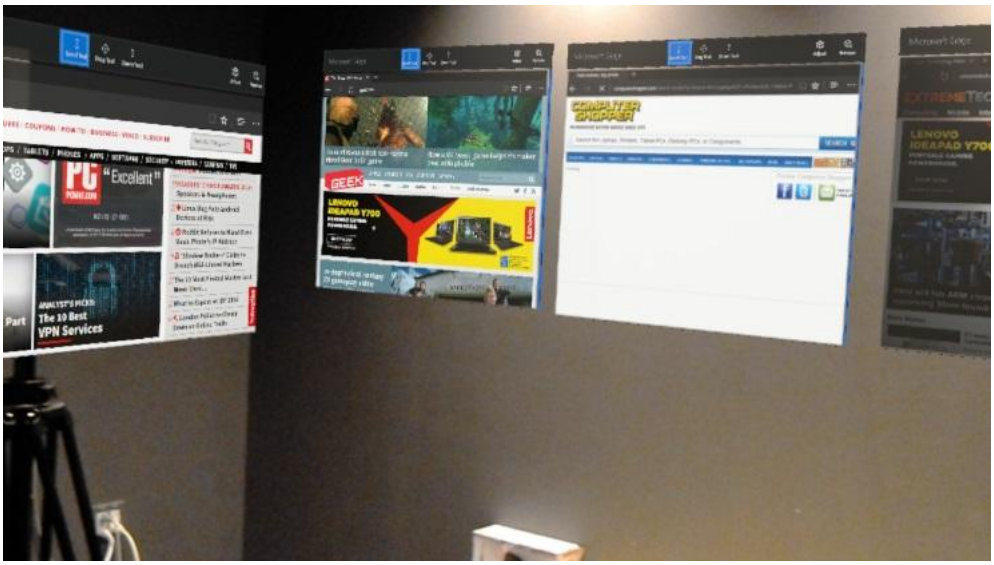
ňovalo si prohlížet model auta z různých úhlů. Díky tomu, že software věděl jaká je velikost vytištěného obrázku, dokázal určit správný scale modelu. Tento přístup k rozšířené realitě se v nadcházejících letech stal populárním a to hlavně pro reklamní účely.

Google Glass

O větší rozšíření povědomí o rozšířené realitě se postaral Google, když v roce 2013 vypustil mezi vývojáře své *Google Glass*. Tyto brýle mají malý průhledný displej viditelný v rohu oka. Displej umožňuje zobrazování informací uživateli aniž by mu výrazně bránily ve výhledu. Cena pro vývojáře byla 1500$ a v průběhu let prošly brýle velkým množstvím úprav. Až v roce 2017 došlo k vydání, ale nakonec jenom pro firmy a ne pro běžné spotřebitele. Jedno z využitích bylo například při operacích. Chirurg měl na sobě brýle a díky brýlím měl neustálý přehled o důležitých životních funkcích operovaného pacienta.

Microsoft HoloLens

Začátkem roku 2015 představila firma Microsoft head-mounted displej obsahující sadu pokročilých senzorů a prostorový zvuk. o rok později byla vydána verze pro vývojáře s cenou 3000$. Zařízení neustále mapuje prostředí a dokáže do něj umísťovat nejrůznější objekty. Operační systém je postaven na platformě *Windows 10* a proto na něm dokážou fungovat i aplikace přímo pro tuto platformu. Kromě ovládání hlasem je hodně využíváno ovládání gesty. Tyto gesta se dělají před zařízením ve vzduchu (třeba pomyslné klikání na myš). Například je možné poměrně plnohodnotně používat prohlížeč nebo *Microsoft Word*. Uživatel před sebou vidí okna s aplikacemi. Tyto okna drží svoji pozici v prostoru, takže je možné mít po levé straně hlavy otevřený *Microsoft Excel* a na druhé straně hlavy mít puštěné video na YouTube jako je to na obrázku 2.1.



Obrázek 2.1: Příklad několika oken různých aplikací umístěných v prostoru z pohledu uživatele používajícího HoloLens

2.2 Aktuální stav rozšířené reality

Výrazná změna nastala, když Apple v roce 2017 představil novou verzi svého operačního systému iOS 11. Ta obsahovala nové SDK nazvané *ARKit*. Podobně vydal Google SDK pro Android zvané *ARCore*. Díky tomu dostaly miliony vývojářů možnost jednoduše vyvíjet aplikace pro tyto dvě platformy. Obzvláště u Applu byl potenciál obrovský, protože z pravidla měsíc po vydání nové verze operačního systému je nainstalován zhruba na 50-ti procentech všech podporovaných zařízení. Vzhledem k tomu, že má Apple přes 1,3 miliardy aktivních zařízení [1] (velká část prodaných zařízení běží na systému iOS), tak počet potenciálních zákazníků se nedal v žádném případě považovat za zanedbatelný.

Jasným využitím pro rozšířenou realitu jsou hry, ale své uplatnění si najde i v dalších oblastech:

- *Medicína* - Studenti medicíny používají anatomicky přesné 3D modely pro účely studia případně celé simulace pro trénování chirurgických zákroků. Existují i zařízení, která dokážou na pacientovi vykreslit přibližnou pozici žil pro přesnější odběr krve.
- *Vzdělání* - Studenti můžou uskutečnit výlet na místa, o kterých se učí, aniž by museli opustit školní lavice. Také vznikají knihy implementující různý obsah v rozšířené realitě.
- *Armáda* - Místo použití drahé vojenské techniky je možné pro výcvik vojáků použít rozšířenou realitu. Využití má i v samotném boji v podobě informací o terénu, přesnější míření nebo identifikování cílů.
- *Architektura* - Zákazník si může prohlédnout a projít se svým budoucím domem ještě před tím než se začne vůbec stavět.
- *Nakupování* - Výběr nábytku a jeho umístění přímo v našem domě, zkoušení oblečení nebo tetování.

Na konci března 2018 vydal Apple aktualizaci iOS 11.3, která přinesla novou verzi ARKitu 1.5. Tato verze přinesla jednu z nejžádanějších funkcí a to detekci vertikálních ploch jako jsou třeba stěny, dveře nebo okna. Další novinkou je detailnější reprezentace detekovaných ploch. V první verzi byly plochy reprezentovány pouze jako obdélníky, ale nyní je možné získat přesnější tvar detekované plochy. Zajímavou novou funkcí je i rozpoznání konkrétních obrazů ve scéně. Podobnou funkcionalitu využilo BMW v již zmiňované reklamě v sekci 2.1. Dále je možné měnit kvalitu videa nebo zapnout *autofocus* na kameře (původně byla kamera zaostřena do nekonečna).

2.3 Existující aplikace

Poté co Apple vydal iOS 11, začala se na App Store (3.3) objevovat spousta zajímavých aplikací využívajících rozšířenou realitu. Podle analytické společnosti SensorTower zaznamenaly aplikace využívající rozšířenou realitu za prvních 6 měsíců od vydání už více než 13 milionů stažení [5]. U některých aplikací pokulhávala kvalita, ale řada aplikací byla kvalitně provedena se zajímavým využitím prvků rozšířené reality.

The Machines

The Machines[1] je jedna z aplikací využívající rozšířenou realitu, která byla představena přímo na vývojářské konferenci Applu společně s iOS 11 a novým SDK ARKit. Konceptem se nejvíce přibližuje hře popisované v této práci, nicméně svou komplexností je ještě dál. Od jiných her využívajících rozšířenou realitu se nejvíce odlišuje možností soupeřit proti skutečným hráčům. Zachází tak daleko, že je možné hrát proti kamarádovi, který je ve stejné místnosti, kdy oba vidí přesně to co druhý pouze z jiného úhlu pohledu. Ve hře je také propracovaný zvukový systém, který ještě více umocňuje zážitek ze hry. Zvuk je například blokován objekty ve hře, takže pokud se hráč schová za kámen, zvuk se od něj odrází a za ním je slyšet pouze tlumeně.

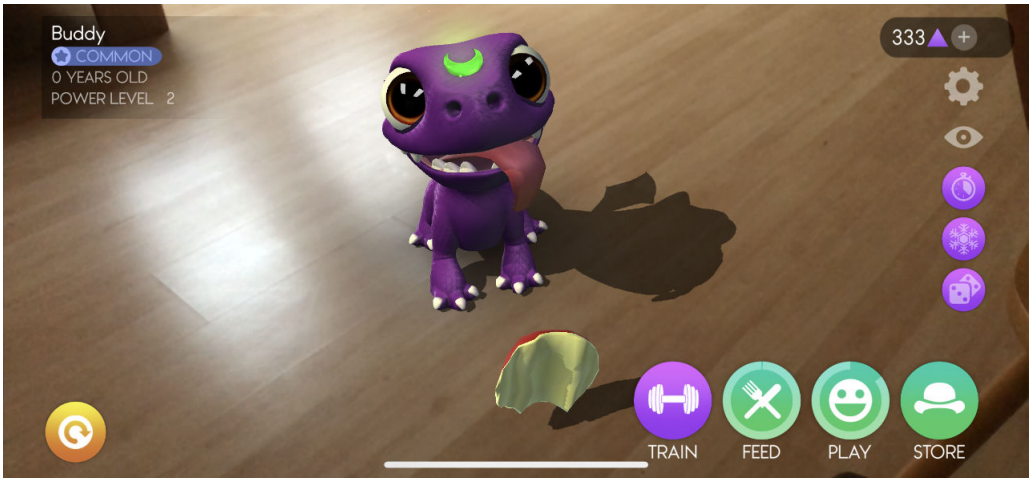
Hra je velice dobře zpracovaná po grafické stránce a snaží se dobře využívat prostoru tím, že různé herní objekty vystupují vysoko od země (obrázek 2.2). Systém samotného umístění je také dobře zpracován a hlavně je kvalitně uživateli vysvětleno co dělat během umísťování. Při prvním spuštění hry uživatele provede tutoriál, který vysvětlí zakladní ovládání a cíl hry.

Obrázek 2.2: Výborně zpracovaná mapa plná malých detailů zasazená do reálného světa ve hře The Machines. Hra je v režimu jednoho hráče proti počítači.

AR Dragon

AR Dragon[2] je aktuálně nejpopulárnější hra využívající rozšířenou realitu v sekci zdarma [5]. Uživatel na začátku hry dostane vajíčko s drakem, který se vylíhne a od té doby je potřeba se o něj starat, trénovat ho, zatímco drak postupem času roste. Při hraní hry je vidět, že je hra schopná detekovat i vertikální plochy, protože když hráč trénuje draka střílením ohnivých koulí na terč, koule se zastaví o zeď, pokud se hráč netrefí. Hra má taky dobře zpracované vrhání stínů do reálného světa (obrázek 2.3).

[1] The Machines https://itunes.apple.com/gb/app/the-machines/id1280682965
[2] AR Dragon https://itunes.apple.com/us/app/ar-dragon/id1270046606?mt=8



Obrázek 2.3: Kvalitní provedení stínů, které jsou poměrně přesně vrženy do reálného světa, ve hře AR Dragon

ARZombi

Hra ARZombi[3] na rozdíl od velké většiny ostatních hojně využívá okolní prostředí, ve kterém se nachází hráč. Před začátkem hry si uživatel namapuje okna a dveře ve svém domě. V těch potom vidí virtuální apokalyptický svět plný zombíků proti kterým musí dům bránit za pomocí různých zbraní.

Oproti hře The Machines (2.3) je umísťování oken a dveří velice špatně implementováno a vysvětleno uživateli a vyžaduje značné úsilí, u nezkušeného uživatele není výjimkou i několik neuspěšných pokusů (obrázek 2.4).

Obrázek 2.4: Ukázka interakce s reálným světem ve hře ARZombi. Na obrázku je vidět, že virtuální dveře jsou posunuty vůči těm skutečným.

[3] ARZombi https://itunes.apple.com/gb/app/arzombi/id1255008041?mt=8

Kapitola 3

Popis využitých technologií

V této kapitole je popsáno, jakým způsobem funguje rozšířená realita v systému iOS. Dále je popsán samotný vývoj a vývojové nástroje pro iOS a další Apple platformy. s tím souvisí i proces vydání aplikace na App Storu. Nakonec je popsáno vytváření her v herním enginu Unity a konkrétní použití rozšířené reality v Unity.

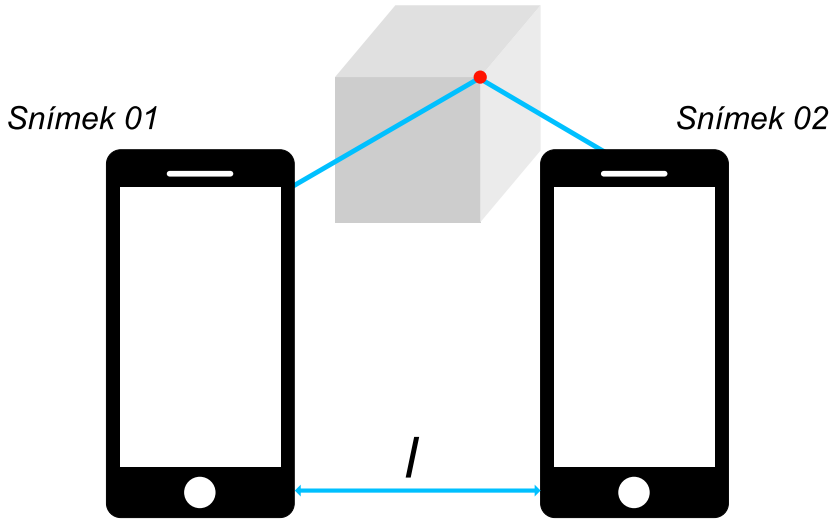
3.1 Rozšířená realita v systému iOS

Rozšířená realita je podporována na všech zařízeních, které mají procesor A9 nebo novější. Na iPhonu X je navíc dostupná funkce pro detailní mapování obličeje za pomocí sady předních kamer a senzorů na tomto zařízení.

ARKit se snaží propojit skutečný svět se světem virtuálním. Aby to dokázal, musí nejdříve vytvořit 3D mapu skutečného světa [6], [9]. Ta se vytváří podobným způsobem jakým lidský mozek dokáže určit přibližnou velikost a vzdálenost objektů, které vidí. Mozek pracuje s obrazem z levého a pravého oka, zná vzdálenost očí od sebe. Potom zná také skutečnou velikost různých objektů a díky tomu dokáže určit přibližnou vzdálenost objektu od oka.

V momentě, kdy začne mapování světa, systém analyzuje snímek z kamery a v něm identifikuje výrazné body. To jsou například hrany stolu nebo přechod mezi podlahou a zdí. Pro každý další snímek se tento proces opakuje. Díky velice přesným senzorům zařízení pro zjištění polohy a natočení zařízení v prostoru lze určit o jakou vzdálenost se kamera posunula a jak se otočila mezi jednotlivými snímky. Toto je znázorněno na obrázku 3.1. Kombinací těchto dat se určí vzdálenost identifikovaných atributů v obraze a jejich propojenost mezi sebou.

Aby ARKit dokázal vytvořit co nejpřesnější reprezentaci skutečného světa, je vhodné, když v obraze identifikuje více výrazných bodů a když se se zařízením pohybuje (ne příliš rychle, to by mělo za následek naopak zhoršení kvality).



Obrázek 3.1: Znázorněno posunutí zařízení v prostoru mezi jednotlivými snímky. V obou snímcích se sleduje červený bod a vzdálenost, o kterou bylo zařízení posunuto je označena *l*.

Nevýhodou výše popsaného způsobu zpracování jsou vysoké požadavky na kvalitu obrazu. Kamery mají v dnešní době stále poměrně velký problém s nedostatkem světla. Toto se projevilo při samotné implementaci a testování hry popisované v této práci. Používala-li se hra v kvalitně osvětlené místnosti, proběhlo analyzování skutečného světa rychle a s velice dobrým výsledkem. Pokud se používala v době, kdy se začínalo stmívat nebo večer, kdy se svítilo umělým osvětlením, několikanásobně se prodloužila doba analyzování a také se zhoršila kvalita reprezentace skutečného světa. K použití bylo daleko méně ploch a ty byly často nepřesně umístěny v prostoru.

3.2 Vývoj aplikací pro platformu iOS

Ke klasickému vývoji aplikací pro platformu iOS (případně macOS, tvOS nebo watchOS) se používá nástroj Xcode. Jeho vývoj zajišťuje přímo Apple a je pouze pro systém macOS, proto není možné vyvíjet aplikace za použití Windows nebo Linux.

Xcode sice podporuje větší množství programovacích jazyků, ale celý editor je výhradně zaměřený na tvorbu aplikací pro platformy Apple, proto je poměrně nepraktický na cokoliv jiného. Ještě do nedávna byl primární vývojový jazyk *Objective-C* (ten bylo možné kombinovat s *C* a *C++*). V roce 2014 Apple představil nový programovací jazyk *Swift*. Od té doby prošel rozsáhlým vývojem a stal se z něj velmi populární programovací jazyk. Již po třech letech se stal používanější než *Objective-C* [11].

Xcode obsahuje celou řadu nástrojů pro vytvoření aplikace. Od vytvoření projektu, editace kódu, tvorby uživatelského rozhraní pomocí WYSIWYG[1] editoru, spuštění aplikace na libovolném zařízení s libovolnou verzí operačního systému pomocí simulátoru, bezdrátová instalace a spuštění na fyzickém zařízení, podepsání aplikace a její následné odeslání do iTunes Connect 3.3.

Součástí jsou i velmi pokročilé nástroje pro ladění aplikace. Jako standard je možnost kdykoliv za běhu aplikace umístit do kódu breakpoint. Až začne aplikace provádět kód,

[1] What You See Is What You Get

kde je breakpoint, dojde k zastavení vykonávání a vývojář si může prohlédnout hodnoty vytvořených proměných (pokud je v proměnné obrázek, lze si ho prohlížet). Navíc lze detailně monitorovat práci s pamětí a najít případné úniky paměti, zátěž procesoru a vliv na baterii zařízení, přenesená a přijatá data přes internet, načítání zdrojů z úložiště zařízení a v neposlední řadě detailní průzkum toho, co dělá grafický procesor v souvislosti s běžící aplikací.

3.3 Potřebné nástroje pro odeslání aplikace na App Store

App Store je digitální platforma vytvořená a udržovaná firmou Apple sloužící pro distribuci aplikací výhradně pro operační systémy iOS, tvOS a macOS. V případě iOS a tvOS je App Store jediná oficiální cesta jak si do zařízení stáhnout aplikace. Ve skutečnosti je ještě jeden způsob, ale využívají ho především firmy, které potřebují aplikace na míru a zároveň chtějí, aby byly distribuovány pouze mezi zaměstnanci firmy a neměl k nim přístup kdokoliv jiný.

Vývojářský účet

Před tím než může vývojář publikovat svoji aplikaci na App Store, musí si vytvořit vývojářský účet u Applu. Tento účet je placený a stojí 99$ na rok. Umožňuje vývojáři publikovat libovolný počet aplikací. Za publikování nejsou žádné další poplatky, pouze v případě, je-li aplikace placená (uživatel za její stažení musí zaplatit). V takovém případě si z každého nákupu Apple nárokuje 30% a vývojáři zbývá 70%. Tento poměr není pevně ustanoven a mění se podle státu a měny. Každý rok je potřeba vývojářský účet předplatit. Pokud to vývojář neudělá, všechny aplikace přestanou být dostupné na App Store.

Po vytvoření účtu musí vývojář vytvořit *App ID*. Je to záznam o aplikaci s unikátním identifikátorem a vážou se na něj nastavení aplikace - jestli využívá Apple Pay, push notifikace, hlasové ovládání pomocí Siri a podobně.

Certifikáty

Následně je nutné vytvořit certifikát obsahující privátní a veřejný klíč [4]. Tento certifikát se používá k podepsání samotné aplikace a identifikuje vývojáře, který aplikaci vytvořil. Na iOS není možné nainstalovat aplikaci, která by nebyla podepsaná. Pro představu by bez podepisování aplikací bylo teoreticky možné vytvořit novou aplikaci s názvem *Facebook* a distribuovat ji jako aktualizaci pro originální aplikaci. Všichni kdo by aktualizaci nainstalovali by používali podvrženou aplikaci, která by je mohla vyzvat k opětovnému příhlášení a tím by vývojář získal přihlašovací údaje k jejich účtům.

Na rozdíl od App ID je možné certifikát použít pro podepsání více aplikací, protože se váže na vývojářský účet. Při vývoji aplikace a jejím testování na zařízeních se používá development certifikát. Aplikace distribuovaná přes App Store musí být podepsaná production certifikátem.

Provisioning profile

App ID a certifikát je potřeba spojit pomocí *provisioning profile* [10], který je vložen do aplikace před každým podepsáním. Při instalaci aplikace na zařízení se nejprve nainstaluje právě *provisioning profile* a systém zkontroluje jestli splňuje určité kritéria, mezi které patří:

- Aplikace je správně podepsaná a privátní klíč použitý k jejímu podepsání sedí k veřejnému klíči.
- Zkontroluje se správnost App ID.
- Nastavení a oprávnění aplikace odpovídají těm, které jsou asociovány s App ID.
- Zařízení na které se instaluje aplikace je v seznamu zařízení (viz. níže).

Pokud některá z těchto podmínek selže, aplikace se nenainstaluje. Při vývoji je navíc nutné, aby *provisioning profile* obsahoval seznam unikátních identifikátorů zařízení. Aplikaci je možné testovat jenom na zařízení obsaženém v tomto seznamu.

Publikace na App Store

Dalším krokem je publikování aplikace na App Store pomocí iTunes Connect. Tady je potřeba vytvořit nový záznam o aplikaci a přiřadit mu App ID. Tím dojde k provázání s nastavením aplikace popsaného výše. K aplikaci se musí doplnit informace, které se budou zobrazovat na App Store. Mezi tyto informace patří název aplikace, kategorie a podkategorie, cena aplikace, screenshoty, popis aplikace, klíčová slova a kontaktní údaje na vývojáře.

Před odesláním aplikace na kontrolu je potřeba nahrát alespoň jednu verzi. To vývojáři zpravidla dělají přímo z vývojového prostředí *Xcode*. Po jejím nahrání dojde k automatickému zpracování a vývojář ji musí přiřadit k verzi aplikace, kterou chce schválit. Aplikaci je možné nejdříve testovat pomocí služby TestFlight 6.2.

Schvalovací proces

Po vyplnění všech vyžadovaných údajů o aplikaci a nahrání buildu aplikace je možné požádat o její schválení. Žádná aplikace se tomuto procesu nevyhne a to ani drobná aktualizace. Všechny aplikace jsou řazeny do fronty a čekají na kontrolu ze strany Applu. Každou aplikaci kontroluje skutečný člověk, proto se doba schválení může vyšplhat i na několik dnů. V případě nalezení nějakého problému dojde k zamítnutí aplikace, vyvojář musí problémy odstranit, v případě, že je problém se samotnou aplikací nahrát i nový build a opět požádat o schválení (aplikace se, až na výjimky, opět zařadí na konec fronty).

Mezi časté důvody zamítnutí aplikace patří [2]:

- Aplikace při schvalování spadla, to znamená okamžité zamítnutí.
- Aplikace nedělá to, co vývojář tvrdí v popisu aplikace.
- Aplikace uživateli nenabízí žádnou přidanou hodnotu - má minimální funkcionalitu nebo obsah případně je zaměřená pouze na velmi specifickou cílovou skupinu.
- Aplikace má velmi nestandardní uživatelské rozhraní.

Po úspěšném schválení je možné aplikaci vydat a tím ji vystavit na App Store, kde bude dostupná ke stažení všem uživatelům App Store.

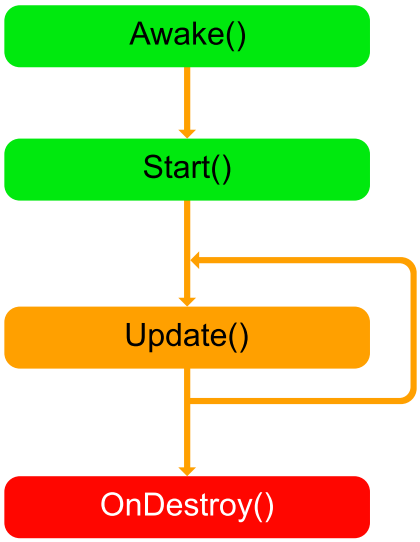
3.4 Unity

Unity je multiplatformní herní engine umožňující tvorbu 2D a 3D her pro desítky různých platforem jako iOS, Android, macOS, Windows, herní konzole nebo WebGL v prohlížečích.

Pro jednodušší hry je často vyžadováno minimální úsilí k tomu, aby hra fungovala na více platformách. Díky svojí rozšířenosti je to skvělý nástroj pro začátečníky, protože má kolem sebe velkou komunitu lidí, kteří tvoří návody pro práci s tímto enginem a pomáhají lidem na různých fórech s jejich problémy a dotazy. i když je Unity častá volba začátečníků, profesionálům nabízí nepřeberné množství pokročilých funkcí a nástrojů a je možné v něm vytvořit hry schopné konkurovat hrám od velkých herních společností. Možnosti grafické kvality, které v Unity aktuálně lze dosáhnout, výborně reprezentuje realtime video vytvořené přímo týmem *Unity Technologies*[2].

Pro editaci zdrojových kódu je možné využít libovolný editor, ale častou volbou bývá *Visual Studio* od firmy Microsoft, protože má přímou podporu Unity. Navíc v roce 2017 vyšla verze tohoto nástroje pro macOS. Jako programovací jazyk se používá *C#*. V minulosti bylo možné použít ještě jazyk *Boo* a do srpna 2017 *JavaScript*.

Každá hra vytvořená v Unity obsahuje alespoň jednu scénu. Scény jsou tvořeny herními objekty *GameObject*, ať už je to kamera, světla, stromy nebo auta. Všechny herní objekty můžou mít libovolný počet komponent, které ovlivňují jejich chování. Všechny třídy dědící z *MonoBehaviour* (v Unity se jim říká skripty) můžou být použity jako komponenty herních objektů. Základní komponentou, kterou mají všechny herní objekty, je *Transform*, definující pozici, rotaci a scale.



Obrázek 3.2: Životní cyklus komponent s často využívanými metodami

Komponenty mají životní cyklus [13] znázorněný na obrázku 3.2, při němž se volají různé metody. Mezi ty nejpoužívanější patří:

- *Awake()* - Volá se okamžitě po tom, co byla vytvořena instance objektu a využívá se pro nastavení daného objektu.
- *Start()* - Ideální pro nastavení, která závisí na jiných komponentách.

[2]Book of the Dead - Unity Interactive Demo - Realtime Teaser https://www.youtube.com/watch?v=DDsRfbfnC_A

- *Update()* - Volá se při každém novém snímku.
- *OnCollisionEnter()* - Dojde-li ke kolizi s jiným herním objektem, zavolá se tato metoda.
- *OnDestroy()* - Volá se těsně před tím, než je skript nebo celý objekt zničen.

Komponenty je možné libovolně přidávat a odebírat z herního objektu. Pro příklad může nastat situace, kdy bychom chtěli hlavnímu hrdinovi ve hře přidat nějakou schopnost po určitou časovou dobu. Hrdinovi můžeme přidat skript popisující tuto schopnost a tím změnit jeho chování. Tento skript se může sám po nějaké době z hrdiny odstranit a tím vrátit jeho chování do původního stavu.

Další důležitou vlastností Unity je možnost ukládat herní objekty do souboru. Těmto objektům se říká *prefab* a slouží jako předloha pro vytváření dalších herních objektů se stejnými vlastnostmi. Konrétně v této práci jsou takto vytvořeni například nepřátelé. Ve hře je jich 9 typů a každý je uložen jako prefab. V každé vlně jde přes mapu několik desítek nepřátel stejného typu. Všichni tito nepřátelé jsou vytvoření jako instance z uložené předlohy (prefab).

Konkrétně pro iOS má Unity určitá omezení když příjde na testování na fyzickém zařízení a vydávání aplikace na App Store. V takové situaci Unity vytvoří nový projekt, který lze otevřít v aplikaci Xcode 3.2. Přes tu je možné hru zkompilovat a nahrát do zařízení nebo ji odeslat na App Store.

Licence

V průběhu let se nabídka licencí různě měnila. Aktuálně se ustálila na modelu umožňujícím používat Unity prakticky každému. Pro malé vývojáře je dostupná Personal licence, která je úplně zdarma, umožňuje využití většiny služeb a funkcí dostupných v Unity. Je podíměna tím, že jedinec nebo společnost nepřekračují příjmy ve výši 100 000$ a při spuštění hry je uživateli zobrazeno logo Unity (tzv. splash screen).

Mezi další nabízené licence patří:

- *Plus* - Příjmy nesmí překročit 200 000$, 20% sleva v Asset Store, možnost upravovat nebo úplně odstranit Unity splash screen, tmavá verze editoru.
- *Pro* - Všechny výhody Plus, navíc má vývojář k dispozici prémiovou podporu. Není limitována objemem příjmů.
- *Unity for Enterprise* - Dostupné pro týmy 21 a více lidí, licence je uzpůsobena konkrétním potřebám dané společnosti.

Asset Store

Podobně jako Apple vytvořil svůj App Store pro distribuci aplikací na svých platformách, vytvořilo Unity službu pro distribuci nástrojů, algoritmů, modelů nebo textur jiným vývojářům. Vývojář může vytvořit užitečný nástroj nebo 3D model s různými animacemi a potom ho pomocí Asset Storu dát ke stažení dalším buďto zdarma nebo za peníze. Vyvojáři potom tyto věci můžou libovolně používat dále ve svých hrách (pokud ve vyjímečných případech není uvedeno jinak).

3.5 Použití ARKitu v Unity

Krátce po představení ARKitu Unity uvolnilo ke stažení plugin[3] vystavující funkce SDK ARKitu přímo v C# kódu hry.

Základem je objekt *UnityARSessionNativeInterface*, jehož existence je v celé hře naprosto unikátní. Pro zkrácení se ve zbytku textu bude používat proměnná *session*, což je instance tohoto objektu získana takto:

```
session = UnityARSessionNativeInterface.GetARSessionNativeInterface();
```

Tento objekt je potřeba nakonfigurovat pomocí *session.RunWithConfig()*. Mezi parametry pro konfiguraci je mimo jiné volba, zdali chceme získávat odhad světelného zdroje nebo chceme mít k dispozici jednotlivé identifikované body při zpracování obrazu. Pokud bysme si chtěli provést detekci ploch individuálně, můžeme ji přes tuto konfiguraci vypnout.

V dalším kroku je potřeba měnit kameru hry podle pohybu zařízení. Vzhledem k tomu, že je nutné toto dělat v každém snímku, umístíme tuto funkcionalitu do *void Update()* metody kamery. Pro získání aktuální pozice a natočení kamery slouží *session.GetCameraPose()*. u kamery je také nutné nastavit projekci. Ta by se měla shodovat s projekcí fyzické kamery na zařízení, jinak by objekty umístěné do reálného světa měly jinou deformaci než objekty ve skutečném světě. K získání projekce kamery slouží *session.GetCameraProjection()*. Kamera na sobě musí mít komponentu *UnityARVideo*. Ta se stará o to, aby se na pozadí vykresloval vstupní obraz z kamery zařízení.

Tyto kroky jsou nezbytné pro základní fungování rozšířené reality ve hře vytvořené v Unity. Dále je možné pomocí *session.GetARAmbientIntensity()* získat z aktuálního vstupu kamery intenzitu a teplotu barvy světla. Tímto docílíme, toho že se bude herní světlo přizpůsobovat světlu ve skutečném světě.

Poslední podstatná funkce pro tuto práci je *session.HitTest()* sloužící k získání detekovaných ploch ve skutečném světe. Jako parametry metoda přijímá bod na obrazovce a typ plochy, který chceme získat. Vrací seznam struktur *ARHitTestResult* daného typu reprezentující místo kde plochu protne paprsek vedený z daného bodu ve směru kamery. Tato struktura mimo jiné obsahuje vzdálenost od kamery k průsečíku a matici reprezentující pozici a rotaci průsečíku.

[3] Unity-ARKit-Plugin https://bitbucket.org/Unity-Technologies/unity-arkit-plugin

Kapitola 4

Návrh hry

Kapitola obecně pojednává o vytvořené hře. O principu na kterém je založena, co vše nabízí a umožňuje hráči. Nastíní způsob fungování a tvoření map a popíše strukturu kódu a jeho fungování. Výsledná hra, která je produktem této práce je k vidění na obrázku 4.1.

Obrázek 4.1: Dva screenshoty ze hry - Výběr a umístění mapy do reálného světa (vlevo), Hra po stisku tlačítka *PLAY* (vpravo)

4.1 Popis hry

Hra je založená na populárním žánru *tower defense*. Na App Store se dá najít nespočet her založených na tomto žánru s různými variacemi a tématikou. Jeho principem je zpravidla menší mapa se začátkem a koncem. Na začátku mapy se v určitém intervalu generují nepřátelé, kteří se snaží dostat na konec mapy. Cesta nepřátel je často předem daná a hráč má předem určené místa, kde může stavět věže, nicméně jsou i hry, kde může hráč stavět prakticky kdekoliv a nepřátelům v réálném čase měnit cestu.

Úkolem hráče je stavět v okolí cesty nepřátel věže, které po nepřátelích střílí a ideálně nenechat projít žádného z nepřátelů na konec mapy. Za zabití nepřátel hráč dostává odměnu většinou v podobě virtuální měny, za kterou si může nakupovat další věže případně vylepšovat již postavené věže. Nepřátelé se většinou generují ve vlnách, tedy se vygeneruje nějaký počet nepřátel a po zabití všech se začnou generovat další, kteří jsou například silnější, rychlejší nebo můžou být odolní vůči nějakému typu útoku.

V této práci je hra založená přesně na tomto principu s tím, že je obohacena o rozšířenou realitu. Hráč si může mapy s nepřáteli a věžemi umístit kamkoliv do skutečného světa a libovolně se kolem ní pohybovat.

4.2 Fungování a možnosti hry

Po spuštění hry se zapne zadní kamera na zařízení a začne se zpracovávat obraz, ve kterém se hledají plochy pro umístění mapy. V momentě kdy se najde vhodná polocha, umístí se na ni mapa (4.1 - obrázek vlevo). Mapa je vždy ve středu obrazovky, takže pohybem zařízení se dá přemístit kdekoliv jinam. Uživatel má na výběr z více map (obrázek 4.1 - bod 4), ale ze začátku je dostupná pouze jedna. Pokud se mu ji podaří dohrát do konce, odemkne se další. Mapu je možné gesty na displeji otáčet a měnit její velikost.

Stiskem stlačítka *PLAY* (obrázek 4.1 - bod 3) se potvrdí pozice mapy a uživatel kolem ní může začít volně chodit, také se znemožní její zvětšování a otáčení. Z obrazovky zmizí UI pro výběr mapy a objeví se herní UI. Po krátké prodlěvě se začnou generovat vlny nepřátel. Každá vlna, kromě poslední, obsahuje 20 nepřátel. Většinou jsou v ní jeden až dva typy nepřátel. Ve dvou vlnách jsou přimýchaní nepřátelé o dvě úrovně težší pro ozvláštnění hry.

Klepnutím na jeden z kamenných podstavců (obrázek 4.1 - bod 3) na mapě dojde k zobrazení výběru věží. Ten se aktualizuje při každé změně, aby dal uživateli vizuálně najevo, jaká věž je mu dostupná a na jakou má nedostatečné finanční prostředky. Výběrem věže dojde k jejímu postavení na označeném podstavci (obrázek 4.1 - bod 7). Pokud se k ní poté nepřítel přiblíží (obrázek 4.1 - bod 8) na dostřelovou vzdálenost, věž po něm začne pálit.

Výběrem podstavce, na kterém již stojí věž, uživatel vyvolá zobrazení okna s informacemi o ní. Uživatel vidí jaké poškození způsobuje, maximální vzdálenost nepřítele, aby po něm věž začala pálit a minimální časovou prodlevu mezi jednotlivými výstřely. Pokud má dostatek peněz, věž může vylepšit, případně ji může prodat. Zde je opět uživateli indikováno, zda na vylepšení má dostatečné zdroje. Podle úrovně vylepšení se mění zabarvení podstavce. Při označení se zabarví do červena, pokud má postavená věž druhou úřoveň, zabarví se do modra a v případě třetí úrovně je zabarven do fialova.

Uživatel má možnost jednou za určitý čas použít plošné kouzlo (obrázek 4.1 - bod 5), které tahem prstu z ikony umístí do mapy. Po startu hry je kouzlo v neaktivním stavu a aktivuje se až po uplynutí 50-ti vteřin. Do mapy dopadne ohnivá koule na dané místo

a způsobí velké poškození nepřátelům, kteří jsou v blízkosti dopadu. Počáteční poloha kouzla je vždy relativní vůči aktuální poloze kamery, aby bylo vždy viditelné.

Po zabití všech nepřátel ve všech vlnách se odemkne následující mapa a uživatel má možnost vrátit se na první obrazovku, tam si vybrat tuto mapu, opět ji umístit do prostoru a začít hrát.

V pravém horním rohu uživatel neustále vidí množství peněz, aktuální číslo vlny, celkový počet vln a počet zbývajících životů (obrázek 4.1 - bod 6). V levém horním rohu je tlačítko pro otevření herních možností. Tady uživatel může hru restartovat, aniž by změnil pozici mapy nebo restartovat hru úplně a vrátit se k výběru mapy, kterou musí znovu umístit.

4.3 Dostupné mapy

Obrázek 4.2: Všechny mapy dostupné ve hře s vyznačenou cestou nepřátel (modrá - začátek, žlutá - záchytné body, červená - konec)

Hra obsahuje tři mapy viditelné na obrázku 4.2. Každá se skládá ze záchytných bodů, které dohromady tvoří cestu nepřátel. Podstavce, na kterých jsou stavěny věže, jsou vždy z velkých placatých kamenů, aby hráči bylo vizuálně jasné, kde může stavět a kde ne. Místa kolem cesty nepřátel jsou většinou vyplněna stromy, menšími kameny případně keři nebo kapradím. V případě druhé a třetí mapy je kolem cesty i voda. Pro lepší efekt není země mapy rovná, ale obsahuje menší kopečky a propadliny.

4.4 Struktura kódu

Celá hra je řízena hlavním objektem *GameManager*, který je vytvořen jako singleton. Obsahuje důležité hodnoty, které se vztahují k celé hře jako je počet životů uživatele, vybraná mapa, počet vln nebo množství peněz. Stará se o chod celé hry. Po tom, co si uživatel vybere mapu, zařídí, aby začala generovat nepřátele. Pokud věž zabije nepřítele, musí tomuto objektu oznámit, že došlo k jeho zabití. *GameManager* přičte uživateli odměnu za zabití a dál se rozhodne, jestli má spustit další vlnu nebo odemknout další mapu případně neudělat nic, pokud jsou v mapě další nepřátelé.

Druhým takovýmto singletonem je *UIManager*. Objekt starající se o celé uživatelské rozhraní hry. Klepne-li například uživatel na podstavec v mapě, oznámí to *UIManageru*

a ten se postará o to, aby se zobrazil výběr věží nebo se zobrazil detail konkrétní věže. Podle stavu hry zobrazuje UI pro umístění a výběr mapy. V průběhu hry zase uživateli zobrazuje herní statistiky a v případě výhry nebo prohry se postará o zobrazení tlačítka pro restart hry nebo výběr nově odmknuté mapy.

Posledním singletonem je *ResourcesManager*. Ten má seznam všech prefabs a textur, které se v průběhu hry používají a k nim přiřazené unikátní identifikátory. Při startu aplikace si všechny věci z tohoto seznamu přednačte. Když je potom někde v aplikaci potřeba načíst nějaký prefab nebo texturu, získá se pomocí identifikátoru z *ResourcesManager*. Důvodem tohoto řešení bylo zaseknutí hry v momentě, kdy se nějaký asset použil poprvé (postavení první věže).

Další významné objekty:

- *Ammo* - Reprezentuje střelu, stará se o pohyb od věže k nepříteli a o zasažení nepřítele.
- *Enemy* - Stará se o chůzi nepřítele po mapě, zobrazení jeho životů a při zabití o jeho odstranění ze scény a připsání odměny hráči.
- *EnemyGenerator* - Generuje všechny různé nepřátele v konkrétní vlně s určitým časovým odstupem.
- *MapPositionController* - Stará se o vše, co se týká mapy - její umístění v reálném prostoru, otáčení a zvětšování.
- *Tower* - Hledá aktivní nepřátele, po jeho výběru se stará o otáčení věže, střílení věže v určitých intervalech a případné vylepšení věže na další úroveň.
- *TowerBase* - Zaznamenává klikání uživatele a na základě toho zobrazuje UI pro výběr věží nebo vylepšení věže. Podle stavu mění barvu objektu a stará se o zničení věže při jejím prodeji.

Kapitola 5

Implementace

V této části je popsána samotná implementace zaměřená na podstatné prvky hry, zejména pak na práci s mapou jako je její umístění do skutečného světa, rotace a změna velikosti. Dále je vysvětleno jakým způsobem se nepřátelé pohybují po mapě, fungování věží a střel. Vysvětleno bude nastavení herních objektů na zakladě kterého se dá jednoduše měnit chování objektů a jaký byl kladen důraz na rozšířitelnost. Nakonec jsou popsány Assety, které byly použity při implementaci hry.

5.1 Umístění mapy do skutečného světa

Drtivá část kódu pro práci s rozšířenou realitou a umisťování mapy je ve třídě *MapPositionController*. Instance této třídy je umístěna jako komponenta na herním objektu, který slouží jako rodič mapy. Tento objekt se vytvoří při spuštění hry ještě před tím než se zapne kamera zařízení. Po vytvoření se přihlásí k získávání aktualizací o každém snímku z kamery a stavu trackování od *UnityARSessionNativeInterface*. Až se objeví nějaký snímek, který má stav *ARTrackingState.ARTrackingStateNormal*, nastaví se interní proměnná *isInitialized = true*, která značí jestli je možné začít snahu o vyhledání vhodného povrchu pro umístění mapy.

Umístění mapy

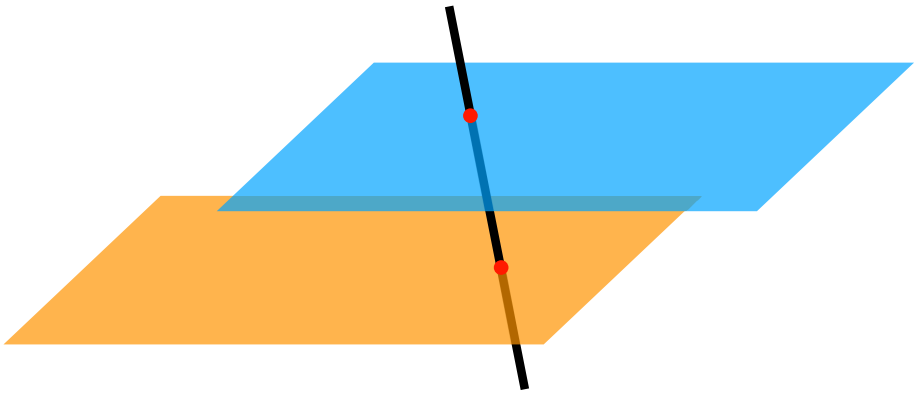
Vzhledem k tomu, že uživatel bude s telefonem neustále hýbat, je nutné, aby se umístění mapy s každým snímkem aktualizovalo. Proto je pro umístění hlavního kódu vhodné použít metodu *void Update()*.

Nejdříve se zkontroluje, jestli je kamera nakloněná dolů, takže nemíří na strop nebo zdi. To se dělá pomocí funkce *Vector3.Dot()*[1], která bere dva parametry, také vektory. První míří směrem dolů a druhý míří dopředu směrem od aktuální pozice kamery. Funkce vynásobí délky těchto vektorů a výslednou hodnotu vynásobí kosinem úhlu mezi těmito vektory. Návratová hodnota je v intervalu <-1; 1>. Pokud je hodnota větší nebo rovna 0, znamená to, že jsou vektory kolmé (kamera směřuje do stěn) nebo míří na opačné strany (kamera směřuje do stropu).

Směřuje-li kamera do země, dojde k vyhledání vhodného povrchu. Vyhledání probíhá v mětodě *ARHitTestResult? NearestHitResult(ARPoint point, ARHitTestResultType type)*. Ta si nejdříve získá všechny možné plochy, které jsou v cestě paprsku vypáleného ze středu

[1] Vector3.Dot() https://docs.unity3d.com/ScriptReference/Vector3.Dot.html

kamery (takže ze středu obrazovky). Vybírá se ten, u kterého došlo k nejmenší změně vzdálenosti od poslední vybrané plochy. Tímto se eliminuje případ, kdy se dvě plochy překrývaly a mapa najednou přeskočila na jinou plochu, která byla v jiné vzdálenosti od kamery. V průběhu implementace byl toto značný problém, protože mapa při přesunování nepříjemně přeskakovala blíž nebo dál od kamery.



Obrázek 5.1: Dvě různě umístěné plochy v prostoru s procházejícím paprskem. Červeně jsou zvýrazněny průsečíky paprsku s plochou.

Toto vyhledání probíhá ve dvou krocích. Nejprve se vyhledávají plochy, které mají konečnou velikost a mapa se na ně umístí jenom v případě, že vypálený paprsek protne tuto plochu (obrázek 5.1). Pokud se žádná taková nenajde, proběhne vyhledání znovu, ale tentokrát se ignoruje jejich velikost a pracuje se s plochami, které se rozpínají do nekonečna (u druhého případu byl výše zmíněný problém s přeskakováním nejznatelnější). Detekce ploch není dokonalá a druhý krok zajistí bezproblémové fungování v případě, kdy mapa byla umístěna na nějaké ploše, ale po přesunutí mimo ni nebylo mapu kam umístit.

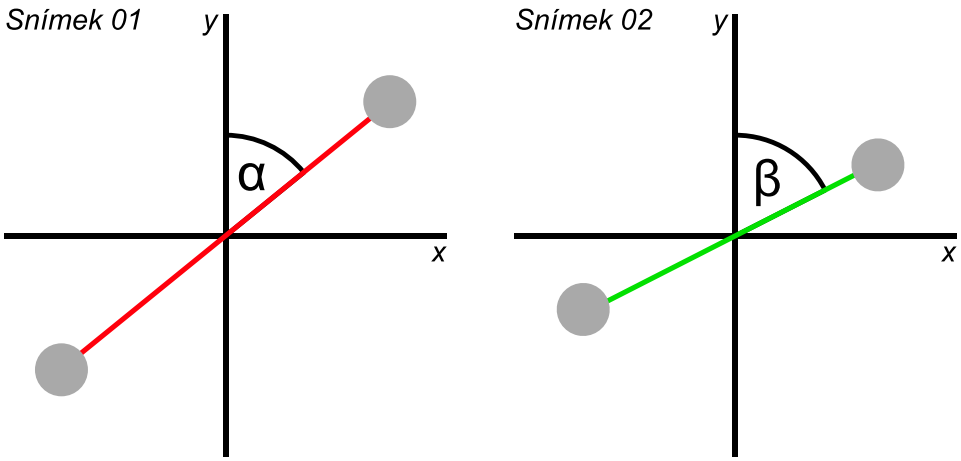
Podle výsledku vyhledání plochy se buďto zobrazí mapa a umístí se na průsečík plochy a paprsku vypáleného ze středu kamery nebo se mapa skryje a místo ní se ve středu obrazovky zobrazí čtverec s pulzující animací, který uživateli naznačuje, že na daném místě nebyla detekována žádná plocha a měl by se zkusit se zařízením rozhlédnout po místnosti.

Zobrazení průběhu trackování

Na začátku hry, kdy si uživatel umísťuje mapu do reálného světa, se zobrazuje aktuální informace o průběhu trackování (obrázek 4.1 - bod 1). Tento průběh se získává z objektu *UnityARCamera*, který je posílán jako parametr metodě *void FrameUpdate()* (viz. 5.1). Tento objekt obsahuje strukturu *ARTrackingState* a *ARTrackingStateReason*. První struktura obsahuje všeobecný stav trackování - jestli je nedostupné, omezené nebo normální. V případě omezeného stavu trackování obsahuje druhá struktura dodatečné informace o důvodu omezení. Jako důvod omezení může být třeba malý počet výrazných atributů v obraze nebo příliš rychlý pohyb se zařízením. Na základě těchto struktur je uživateli zobrazována informace o stavu trackování.

Rotace a zoomování

Při umisťování mapy s ní uživatel může libovolně otáčet a zoomovat. i tuto část bylo nejvhodnější řešit při každém snímku a tedy v metodě *void Update()*.



Obrázek 5.2: Změna polohy prstů mezi dvěma snímky. Prsty jsou spojeny úsečkou. Každá má určitou délku a úhel otočení.

Tato funkce je aktivována pouze pokud uživatel umístí na obrazovku dva prsty a jejich vzdálenost od sebe je větší než *50* bodů. Nejprve se při prvotní detekci dvou prstů vytvoří vektor reprezentující úsečku mezi prsty (červená čára na obrázku 5.2) a uloží se jako poslední známý. V každém dalším volání metody *void Update()* se provedou tyto kroky:

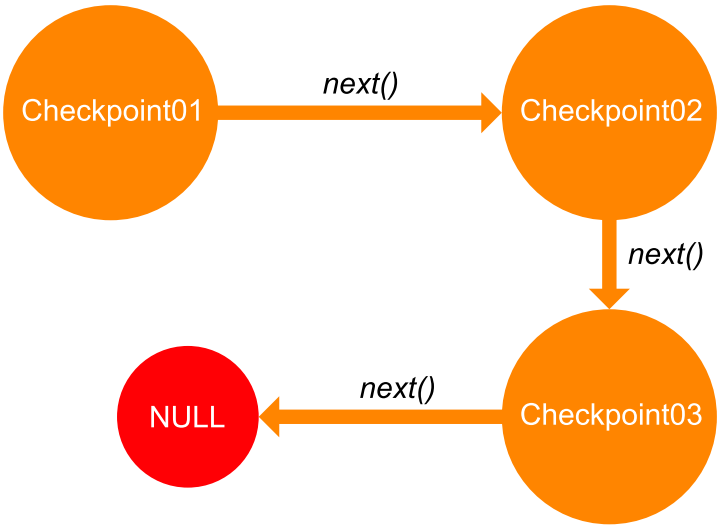
1. Vytvoří se vektor reprezentující aktuální polohu prstů (modrá čára).
2. Spočítá se úhel mezi aktuálním a posledním známým vektorem (rozdíl mezi úhlem α a β).
3. Pomocí funkce *Vector3.Cross()* se vytvoří nový vektor, který je kolmý k aktuálnímu a poslednímu vektoru (je kolmý na obrazovku zařízení) Podle toho, jestli směřuje do obrazovky nebo z ní vystupuje se pozná, na kterou stranu se prsty otáčí.
4. Podle směru otáčení se k aktuální rotaci mapy přičte nebo odečte úhel spočítaný v bodě 2.
5. Aktuální vektor se nastaví jako poslední známý.

Zoomování mapy je o poznání jednodušší. Spočítá se vzdálenost mezi prsty při posledním a aktuálním volání metody *void Update()*. Rozdíl těchto hodnot se vynásobí konstantou *-0.0003* sloužící ke kontrole rychlosti zoomování (na této hodnotě se konstanta ustálila v průběhu testování jako nejideálnější). Výsledná hodnota se přičte k vektoru reprezentujícího zvětšení mapy. Pokud je nově vzniklý vektor v rozmezí minimálního a maximálního zvětšení, přiřadí se jako nové zvětšení mapě.

5.2 Mechanika věží a nepřátel

Nepřátelé

Každá mapa má několik objektů, které reprezentují cestu, po které jdou nepřátelé. Tyto objekty mají komponentu *Checkpoint*, která ukazuje na další bod v cestě. Tento postup byl zvolen kvůli jednoduché rozšiřitelnosti. Pokud by došlo k přidání nové mapy, tak u ní pouze stačí označit, který bod je počáteční. Po vytvoření nepřítele se mu přiřadí počáteční bod s komponentou *Checkpoint*. Dojde k umístění nepřítele na tento bod a spustí se chůze. Z komponenty *Checkpoint* se vezme další bod v cestě a objekt nepřítele se posunuje k tomuto bodu. Až se k němu přiblíží, opět se z něj vezme další bod v cestě. Toto se opakuje do doby, kdy už není další bod, ke kterému by se mohl nepřítel přibližovat (viz. obrázek 5.3). Až nastane tato situace, znamená to, že došel na konec cesty. Objekt nepřítele animací zmizí a hráči se ubere jeden život za to, že mu nepřítel prošel celou mapou. Pokud by takto prošli 3 nepřátelé, hra se ukončí a uživateli se zobrazí obrazovka, která ho informuje o neúspěchu s možností hru restartovat.



Obrázek 5.3: Reprezentace cesty nepřátel pomocí řetězu bodů. Každý bod ukazuje na následující. Poslední bod ukazuje na *NULL* - tedy nikam.

Věže

Mechanika věží je značně komplikovanější než mechanika nepřátel. I v tomto případě hlavní funkčnost zajišťuje metoda *void Update()*, která je prováděna při každém snímku.

Nejprve se zkontroluje, zda má věž cíl na který může střílet. V případě, že nemá žádný cíl, cíl je neaktivní (nepřítel byl zabit) nebo je vzdálenost cíle od věže větší než maximální dostřelová vzdálenost věže, vyhledá se nový cíl.

Vyhledání iteruje všechny nepřátele, kteří jsou aktuálně na mapě. Pokud se najde takový, který je aktivní a zároveň jeho vzdálenost spadá do dostřelové vzdálenosti věže, použije se jako nový cíl pro věž.

Pohyb věže

Je potřeba, aby se věž postupně otočila za svým cílem pokud směřuje jiným směrem. Každá věž má dvě části. Jedna se otáčí kolem osy y a druhá kolem osy x. Pro tyto dvě části se spočítá nová rotace tak, aby směřovaly k cíli a za použití funkce *Quaternion.Lerp()*[2], která interpoluje mezi původní rotací a požadovanou rotací, se otočí o určitou část blíže k požadované rotací. Díky tomu, že se tato část kódu provádí při každém snímku, vznikne pro uživatele plynulá animace otáčení věže za svým cílem.

Až je věž zamířena na cíl, zkontroluje se, jestli čas, který uplynul od posledního výstřelu, je větší než minimální čas, po kterém může věž znovu střílet. Jestliže podmínka platí, vytvoří se objekt s komponentou *Ammo* reprezentující střelu, předá se mu cíl a čas posledního výstřelu se vynuluje.

Střela

Střela poté převezme řízení výstřelu. Podobně jako u pohybu nepřátel se při každém snímku posune o něco blíže ke svému cíli. Když se k němu dostane, vytvoří se objekt s částicovým systémem, rerpezentující zásah nepřítele. Zároveň se nepříteli předá poškození, které má zásah způsobit a také typ poškození (obyčejné, zásah ohněm, zásah mrazem).

Nepřítel odečte poškození od zbývajícího počtu životů. V případě že jeho životy klesnou na nulu, deaktivuje se a spustí se animace umírání (animace je jiná než animace v případě, kdy nepřítel dojde na konec mapy).

5.3 Nastavení herních objektů a rozšířitelnost

Při implementac kladen důraz na rozšířitelnost hry, aby bylo jednoduché přidávat další obsah jako jsou mapy, věže, nepřátele a vlny nepřátel. Proto jsou všechny hodnoty, které definují vlastnosti těchto objektů vyňaty z kódu a umístěny do speciálního objektu typu *ScriptableObject*. Tento objekt slouží jako předloha a vytváří se podle něj další s již konkrétními hodnotami. Takto vytvořený objekt je uložený do souboru a přiřazuje se k entitě ve hře, která si z něj načítá potřebná data.

```
public class EnemyConfig: ScriptableObject
{
    public float enemySpeed = 1.0f;
    public float enemyRotationSpeed = 0.1f;
    public int health = 100;
    public int reward = 30;
    public float enemyScale = 0.6f;
    public bool flipped = false;

    public void description() {}
}
```

Výpis 5.1: Struktura konfiguračního souboru pro nepřátele *EnemyConfig*

Lze pomocí něj nastavit vlastnosti jako rychlost pohybu, rychlost otáčení, zdraví nepřítele nebo odměna, jakou hráč dostane za jeho zabití.

[2]Quaternion.Lerp() https://docs.unity3d.com/ScriptReference/Quaternion.Lerp.html

Všechny typy nepřátel ve hře mají vlastní kofigurační soubor podle předlohy *Enemy-Config* s různě nastavenými hodnotami. Pro nový typ nepřítele potom stačí vytvořit nový konfigurační soubor.

Další výhoda tohoto přístupu je možnost snadné editace. Konkrétně u tohoto žánru hry je nutné aby byly všechny vlastnosti herních objektů dobře vybalancovány, protože špatným nastavením atributů by hra po několika vlnách mohla začít být extrémně těžká nebo naopak nepřiměřeně lehká a tím by pro uživatele přestala být zajímavá. Proto při snaze tyto vlastnosti vybalancovat velmi pomůže, když se dají jednoduše upravovat z jednoho místa a nemusí se vůbec zasahovat do kódu.

Mimo jiné se na rozšiřitelnost myslelo i u uživatelského rozhraní. Tlačítka pro výběr mapy nebo výběr věže jsou umístěny v objektu, který si upravuje velikost podle obsahu, takže se s každým přidáním nové mapy nebo věže a k nim odpovídajících tlačítek roztáhne. Tento objekt je umístěn ve *ScrollRect*[3]. Díky tomu se s tlačítky dá scrollovat v případě, že je jich více než se vleze na obrazovku.

5.4 Použité Assety

Při vytváření hry bylo použito několik assetů nakoupených přes Asset Store 3.4. Díky tomu jsem se mohl daleko více soustředit na samotnou implementaci hry a nemusel se třeba učit vytvářet a animovat 3D modely, což by bylo nad rámec této práce. Navíc to umožnilo udělat hru na stávající grafické úrovni.

Jedním z použitých Assetů je Blend Mode Shader 2D[4] umožňující splynutí obrazu s jeho pozadím za použití nějakého efektu. Tyto efekty mají imitovat ty, které jsou nabízeny například v aplikaci Photoshop (multiply, overlay, darken, lighten, ...). Konkrétně byl použit na terč sloužící k určení dopadu ohnivého kouzla.

Dalším Assetem je Mesh Terrain Editor Free[5]. Tento Asset slouží pro tvorbu 3D terénu podobně jako nástroj, který je přímo v Unity - Terrain. Rozdíl je ten, že tento nástroj vytváří terén jako mesh (změny v terénu jsou přímo prováděny na 3D modelu), kdežto Terrain v Unity využívá 2D mapu. To zapříčiňuje, že terénu vytvořenému pomocí Terrain není možné měnit velikost.

Dalšími použitými Assety jsou:

- *Top-Down Simply forest*[6] - Modely stromů, kamenů, květin a dalších použitých pro vytvoření map.
- *Sci-fi Turrets (cannon)*[7] - Modely věží.
- *Characters Creature Pack*[8] - Modely všech nepřátel ve hře.

[3] ScrollRect https://docs.unity3d.com/ScriptReference/UI.ScrollRect.html

[4] Blend Mode Shader 2D https://assetstore.unity.com/packages/vfx/shaders/blend-mode-shader-2d-93186

[5] Mesh Terrain Editor Free https://assetstore.unity.com/packages/tools/terrain/mesh-terrain-editor-free-67758

[6] Top-Down Simply forest https://assetstore.unity.com/packages/3d/environments/fantasy/top-down-simply-forest-61300

[7] Sci-fi Turrets (cannon) https://assetstore.unity.com/packages/3d/props/weapons/sci-fi-turrets-cannon-69615

[8] Characters Creature Pack https://assetstore.unity.com/packages/3d/characters/creatures/characters-creature-pack-50612

- *DOTween*[9] - Vytváření animací v kódu.

Všechny obrázky použité pro uživatelské rozhraní potom vytvořil na základě mého návrhu rozložení můj kamarád Tomáš Svozil, který se ve volném čase věnuje herní grafice.

[9]DOTween http://dotween.demigiant.com

Kapitola 6

Testování

V poslední kapitole je pospáno, jakým způsobem byla hra testována. Jsou zde popsány nástroje použité k testování, testování na uživatelích, zpětná vazba od těchto uživatelů, vyhodnocení získaných dat, zapracované změny na základě těchto dat a v neposlední řadě možnosti jak hru vylepšit a rozšířit.

6.1 Unity Analytics

Unity nabízí vlastní nástroj pro sběr dat z her a jejich následnou analýzu - Unity Analytics[1]. Díky tomuto nástroji je možné sledovat různé oblasti hry a kontrolovat jak se hráčům daří, kde mají problémy, případně které části hry jsou příliš jednoduché.

Unity Analytics jsou zdarma dostupné i pro uživatele se základní licencí Unity Personal. Proto jsem se rozhodl je do hry implementovat za účelem získání většího množství dat z testování hry.

Hra odesílá následující analytická data:

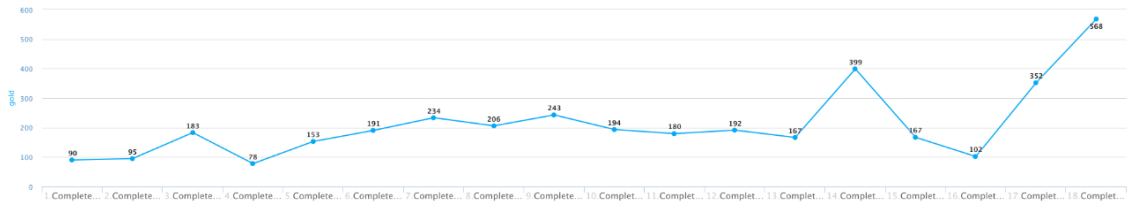
- *Spuštění hry* - Odesílá se po stisknutí tlačítka *PLAY*, obsahuje číslo mapy.
- *Zvládnutí vlny nepřátel* - Odesílá se po zabití všech nepřátel v dané vlně. Obsahuje číslo mapy, číslo vlny, a množství peněz
- *Dokončení mapy* - Odesílá se po zabití všech nepřátel ve všech vlnách. Obsahuje číslo mapy a množství peněz.
- *Restart hry* - Odesílá se vždy, když uživatel restartuje mapu. Obsahuje číslo mapy, číslo vlny, množství peněz a počet životů.
- *Uživatel prohrál* - Odesílá se poté, co uživatel ztratí všechny životy a prohraje. Obsahuje číslo mapy, číslo poslední vlny a množství peněz.

Mezi další odesílaná data patří informace o použití ohnivého kouzla, postavení věže a její typ, vylepšení věže případně prodej věže. Také lze zjistit počet uživatelů, kteří hru spustili, v jednotlivých dnech.

Z těchto dat lze vyčíst, které vlny dělaly uživatelům největší problém. Pokud má nějaká vlna velké množství uživatelů, kteří ji dohráli a poté následuje vlna, kde je velký propad, dá se předpokládat, že tato vlna je pro uživatele příliš obtížná. Naopak u vln, kde uživatelé

[1] Unity Analytics https://unity.com/solutions/analytics

většinou nemají problém, lze podle množství peněz po dané vlně vyčíst, jestli není až příliš jednoduchá.



Obrázek 6.1: Graf průměrného množství peněz, které měli uživatelé na konci každé úspěšně dokončené vlny

V průběhu testování byly tyto data využity pro dodatečné balancování herních statistik, takže někde došlo ke snížení atributů nepřátel, u jiných vln zase k jejich zvýšení. Také se různě upravovaly ceny věží a jejich vylepšení a měnila se účinnost ohnivého kouzla v jednotlivých vlnách. Cena věží se upravovala hlavně na základě toho, jak se uživatelům dařilo dokončit jednotlivé vlny a kolik měli na konci vlny peněz jak je vidět na obrázku 6.1.

6.2 TestFlight

K distribuci hry uživatelům za účelem testování byla použita služba TestFlight. Tato služba je součást iTunes Connect 3.3. Dříve byla služba dostupná i pro distribuci testovacích buildů pro platformu Android, ale poté co ji odkoupil Apple, tuto možnost zrušil.

TestFlight umožňuje interní testování a externí testování. V případě interního testování musí být všichni uživatelé přidáni v iTunes Connect. Přes email uživatelům příjde pozvánka k testování, kterou musí na svém iOS zařízení potvrdit. Poté dostanou kód, který je potřeba zadat v aplikaci TestFlight[2]. Po jeho zadání je možné z této aplikace testovanou aplikaci kdykoliv stáhnout a při každém vydání nové verze i aktualizovat.

Externí testování nevyžaduje, aby uživatelé byli přidáni v iTunes Connect, ale vyžaduje schválení první verze aplikace Applem podobně jako před vydáním aplikace na App Store 3.3. Interní testování je limitováno pro maximální počet 25 uživatelů, kteří si mohou aplikaci nainstalovat, kdežto externí testování umožňuje až 10 000 uživatelů.

6.3 Dotazník a názor uživatelů

Všichni uživatelé, kteří hru testovali dostali dotazník pro získání zpětné vazby. Dotazník obsahoval otázky na pohlaví a věk uživatelů, jejich zkušenosti s typem hry tower defense, názor na grafické zpracování hry, přívětivost a srozumitelnost uživatelského rozhraní, obtížnost hry a zda je hra jako celek bavila. Nakonec měli uživatelé možnost napsat své připomínky ke hře, nalezené chyby případně návrhy na zlepšení.

[2]TestFlight na App Store https://itunes.apple.com/cz/app/testflight/id899247664?mt=8

Vyhodnocení

Konečný počet uživatelů, kteří se zúčastnili testování byl 11. Většina z nich byli muži - celých 81,8% (graf B.1). 63,8% testujících (graf B.2) se věkem pohybovalo mezi 22 až 30-ti lety. Druhé velké skupině bylo 18 až 22 let. Zkušenosti s hrami typu tower defense mělo 54,6% a polovina z nich je dokonce hraje často (graf B.3). Velice pozitivní byl ohlas na grafiku. Celých 63,6% odpovědí (graf B.4) bylo "Velice pěkná"a zbytek s ní neměl problém. Uživatelské rozhraní by evidentě potřebovalo vylepšit, protože jenom 27,3% uživatelům přišlo intuitivní (graf B.5). Zbylé otázky, uživatelské připomínky a přesné výsledky dotazníku jsou k nalezení v příloze B.

Jedna z častých připomínek uživatelů byla nevyváženost a přílišná obtížnost vln. Proto podle dat získaných pomocí Unity Analytics (6.1) došlo k úpravám v obtížnosti jednotlivých vln i dalších aspektů hry.

Mezi další přípomínky uživatelů, které byly vzaty v potaz a do hry doplněny byla příliš dlouhá prodleva než se podařilo mapu někam umístit (hlavně v horších světelných podmínkách). Na základě toho byla do hry doplněna detekce dalšího typu plochy a to *ARHitTestResultTypeEstimatedHorizontalPlane*, která hledá libovolnou plochu kolmou ke gravitaci. K jejímu nalezení dojde v řádu stovek milisekund, v horším případě několik sekund, což je podstatné zlepšení oproti původnímu fungování. Uživatelům se také nelíbilo, že při umísťování kouzla nebylo úplně jasné, kam nakonec kouzlo dopadne. Terč znázorňující místo dopadu byl totiž jenom na obrazovce přímo pod uživatelovým prstem. Ná základě toho byl terč přesunut do mapy a pozice prstu na obrazovce se přepočítává do pozice v mapě. Tím je místo dopadu naprosto jasné, zároveň je velikost terče upravena tak, aby znázorňovala dostah kouzla a uživatel ví, které nepřátele ještě zasáhne a které už ne.

Uživatelé také našli větší množství chyb. Dvě z nich úplně znemožňovaly hru dohrát. V určitých situacích nastalo zneaktivnění tlačítka pro vylepšení věže i přesto, že věž měla dostupné vylepšení a uživatel na něj měl dostatek peněz. V jistých případech zase nedošlo ke korektnímu ukončení vlny po zabití všech nepřátel a další vlna se už nespustila. Mezi menší chyby patřilo občasné střílení věží jiným směrem než byl nepřítel nebo střílení věží do nepřátel, kteří již byli mrtví.

Výkon

Jasně největším a nejčastěji zmiňovaným problémem byl výkon hry a zahřívání telefonu. Uživatelé si stěžovali na výrazné trhání hry při větším množství nepřátel a věží v mapě a také na velké zahřátí telefonu při hraní.

Při spuštění hry pomocí Xcode (3.2) přímo do zařízení lze vidět základní informace o vytížení procesoru, grafiky a operační paměti v průběhu hraní hry. Před optimalizací se vytížení procesoru pohybovalo přes *200%* a občas se vyšplaholo i nad *250%*. Využití operační paměti se stabilně drželo na *300MB*.

Unity nábízí velice užitečný nástroj Unity Profiler[3] pro důkladné zkoumání výkonu her. Za jeho využití jsem zjistil, že nejvýraznější dopad na výkon má VSync, který synchronizuje frame rate hry s frame ratem displeje a zabraňuje tím artefaktům v obraze, kdy je obraz v některých místech horizontálně posunut. Jeho vypnutím došlo k výraznému poklesu ve vytížení zařízení.

[3]Unity Profiler https://docs.unity3d.com/Manual/ProfilerWindow.html

Další optimalizací bylo odstranění světel z kouzel, které střílí věže. Tím docházelo k tomu, že kromě hlavního světla bylo ve scéně i několik dalších světel. Unity doporučuje mít ve scéně ideálně jenom jedno světlo [14].

U map došlo k zapnutí atributu *Static*. Unity toto doporučuje nastavit u nepohybujících se objektů, aby u nich mohly proběhnout interní optimalizace.

Nakonec proběhla optimalizace velikosti textur. Jako výchozí nastavení jsou textury v jejich plném rozlišení. Například u nepřátel jsou textury o velikosti 2048x2048 pixelů. Vzhledem k jejich výsledné velikosti na obrazovce je toto zbytečně přehnané a v rámci optimalizace se použila poloviční velikost textur.

Po všech těchto optimalizacích byla odezva uživatelů velice pozitivní. Došlo k výraznému snížení zahřívání telefonu, zlepšila se plynulost hry a po grafické stránce uživatelé nezaznamenali žádné zhoršení. Spuštění optimalizované verze na zařízení pomocí Xcode ukázalo snížení vytížení procesoru na *60%* až *80%* a využití operační paměti se snížilo z *300MB* na *200MB*.

6.4 Možnosti vylepšení a rozšíření

Jako jedno z největších rozšíření, které se nabízí, je podpora operačního systému Android a jeho knihovny pro rozšířenou realitu ARCore. Hra byla navržena s myšlenou snadné rozšiřitelnosti, proto by rozšíření na Android vyžadovalo převážně zásahy ve třídě *MapPositionController*, který přímo pracuje s funkcemi ARKitu a pro zbytek kódu tyto funkce zaobaluje.

Jedna z navrhovaných funkcí od testujících uživatelů byla možnost celou hru tlačítkem zrychlit a kdykoliv zase zpomalit. Tato funkce by se dala dobře implementovat pomocí *Time.timeScale*[4].

Po vzoru hry The Machines (2.3) by bylo dobré uživatele trochu více provést hrou. Jak celým procesem umístění mapy, tak i rychle uživateli předvést jak postavit věž nebo použít ohnivé kouzlo.

Další funkce, které by mohly přispět k lepší hratelnosti jsou zobrazení odpočtu do spuštění každé vlny a zobrazování dostřelu věží při jejich výběru. Po grafické stránce by se mohlo celé uživatelské rozhraní rozšířit o animace a mohl by se přidat částicový efekt při dopadu ohnivého kouzla na zem.

[4]Time.timeScale https://docs.unity3d.com/ScriptReference/Time-timeScale.html

Kapitola 7

Závěr

Cílem práce bylo navrhnout a implementovat hru v rozšířené realitě pro platformu iOS, což zahrnovalo nastudování rozšířené reality jako takové, její principy a základy jejího fungování. Dále bylo zapotřebí celkově se seznámit s platformou iOS a vývojem aplikací pro ni. Toto obsahovalo mimo jiné vytvoření vývojářského účtu u firmy Apple a následné zaregistrování hry, aby bylo možné dále vytvářet všechny potřebné náležitosti pro iOS aplikaci jako je například certifikát k podepsání.

Poté bylo zapotřebí seznámit se s multiplatformním herním enginem Unity sloužícím k tvorbě her pro velké množství platforem včetně iOS. Tady bylo důležité prostudovat základní principy práce a tvorby her v tomto nástroji a následně způsoby jakými se v tomto nástroji tvoří hry využívající rozšířenou realitu specificky pro platformu iOS.

Výsledkem je funkční hra typu tower defense obsahující tři mapy, tři typy věží (každá má tři úrovně vylepšení) a 9 typů nepřátel. Každá mapa obsahuje 18 vln skládajících se z různých typů nepřátel. Různé 3D modely využité v této hře byly zakoupeny na Asset Store. Výsledná hra funguje na všech iOS zařízeních, které mají procesor A9 a novější. Do této skupiny spadá iPhone 6S a novější, všechny verze iPadu Pro případně všechny další iPady představené v roce 2017 a později.

Když se aplikace dostala do pokročilejšího stádia implementace, započalo její testování. Nejdříve se jednalo o řízené testování s malým počtem lidí. Na základě sledování jejich interakce s hrou se měnily a vylepšovaly některé aspekty hry. Následně byla hra testována větším počtem uživatelů, kteří po dohrání byli požádáni o vyplnění dotazníku.

Díky tomuto testování byly odhaleny menší i větší chyby hry, případně nedostatky zhoršující kvalitu uživatelské zkušenosti. Pomocí analytických nástrojů implementovaných do hry bylo vidět, jaké části hry jsou příliš obtížné nebo naopak příliš lehké. Tyto data přispěly k balancování hry tak, aby byla pro uživatele pokud možno co nejzajímavější.

Poslední částí bylo odladění výkonu hry. Velké množství testovacích uživatelů si stěžovalo na přílišné zahřívání telefonu a na časté sekání hry. Důkladným zkoumáním pomocí dostupných nástrojů v Unity byly odhaleny atributy, které nejvíce ovlivňovaly výkon. Výsledkem následné optimalizace byl až překvapivý pokles ve vytížení procesoru v zařízení.

Již od začátku bylo myšleno na snadnou rozšířitelnost kódu, díky tomu by bylo poměrně snadné do hry přidávat další obsah jako jsou věže, mapy nebo nepřátelé. Určitě největším přínosem by bylo rozšíření na platformu Android pro získání většího počtu uživatelů. Další možnosti rozšíření jsou popsány v poslední kapitole. Zastávám názor, že kdyby se tato rožšíření implementovala, aplikace by si mohla najít své místo na App Store.

Literatura

[1] Apple, Inc.: *Apple Reports First Quarter Results*. [Online; navštíveno 04.04.2018].
URL https://www.apple.com/newsroom/2018/02/apple-reports-first-quarter-results/

[2] Apple, Inc.: *Common App Rejections*. [Online; navštíveno 13.04.2018].
URL https://developer.apple.com/app-store/review/rejections/

[3] Barfield, W.: *Fundamentals of Wearable Computers and Augmented Reality, Second Edition*. CRC Press, Jul, 2015, ISBN 978-1482243505.

[4] Kollbach, T.: *Inside Code Signing*. [Online; navštíveno 08.04.2018].
URL https://www.objc.io/issues/17-security/inside-code-signing/

[5] Nelson, R.: *ARKit-only Apps Surpass 13 Million Downloads in First Six Months, Nearly Half from Games*. [Online; navštíveno 01.04.2018].
URL https://sensortower.com/blog/arkit-six-months

[6] Piao, J.-C.; Kim, S.-D.: *Adaptive Monocular Visual-Inertial SLAM for Real-Time Augmented Reality Applications in Mobile Devices*. MDPI, Nov, 2017, [Online; navštíveno 28.03.2018].
URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5712971/

[7] Rosenberg, L. B.: *Virtual fixtures: Perceptual tools for telerobotic manipulation*. Sep, 1993, [Online; navštíveno 02.04.2018].
URL https://ieeexplore.ieee.org/document/380795/

[8] Schmalstieg, D.; Hollerer, T.: *Augmented Reality: Principles and Practice*. Addison-Wesley Professional, Jun, 2016, ISBN 978-0-321-88357-5.

[9] Shelley, M. A.: *Monocular Visual Inertial Odometry on a Mobile Device*. Aug, 2014, [Online; navštíveno 29.03.2018].
URL https://vision.in.tum.de/_media/spezial/bib/shelley14msc.pdf

[10] Sriskantha, H.: *A Beginner's Guide to iOS Provisioning Profiles*. [Online; navštíveno 08.04.2018].
URL https://blog.theodo.fr/2017/02/a-beginners-guide-to-ios-provisioning-profiles/

[11] Stack Overflow: *Developer Survey Results 2017*. [Online; navštíveno 08.04.2018].
URL https://insights.stackoverflow.com/survey/2017/#technology-programming-languages

[12] Sutherland, I. E.: *A HEAD-MOUNTED THREE-DIMENSIONAL DISPLAY*. Dec, 1968, [Online; navštíveno 05.05.2018].
URL http://90.146.8.18/en/archiv_files/19902/E1990b_123.pdf

[13] Unity Documentation: *Execution Order of Event Functions*. [Online; navštíveno 03.05.2018].
URL https://docs.unity3d.com/Manual/ExecutionOrder.html

[14] Unity Documentation: *Optimizing graphics performance*. [Online; navštíveno 26.04.2018].
URL https://docs.unity3d.com/Manual/OptimizingGraphicsPerformance.html

Příloha A

Obsah DVD

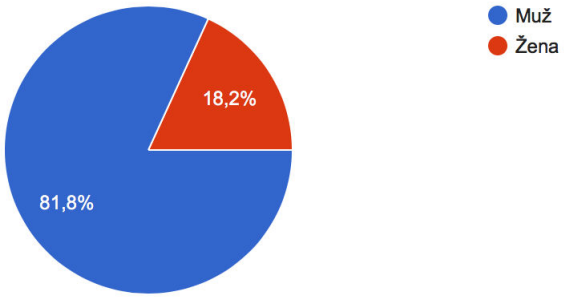
Přiložené DVD obsahuje následující:

- *defender_ar* - Všechny zdrojové kódy výsledné aplikace.
- *bakalarska_prace.pdf* - Tato práce ve formátu PDF.
- *bakalarska_prace_latex* - Všechny zdrojové kódy této práce.
- *defender_ar_gameplay.mp4* - Video ukazující umístění, otočení a zvětšení mapy a poté úspěšné dokončení všech 18-ti vln první mapy.
- *README* - Popis práce s přiloženými soubory.

Příloha B

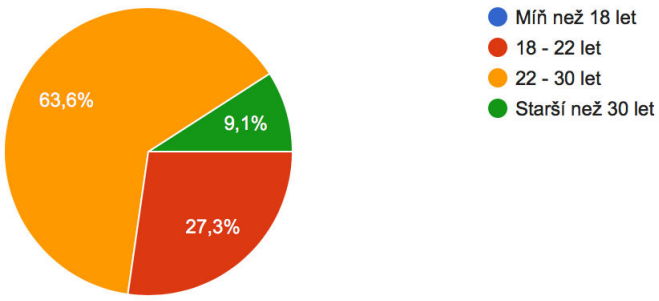
Výsledky dotazníku

Pohlaví



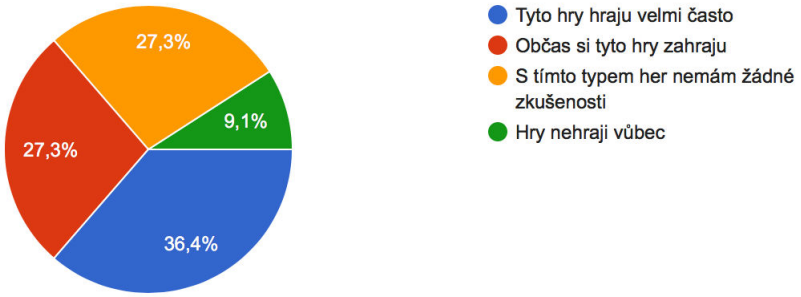
Obrázek B.1: Výsledek dotazníku - Pohlaví

Věk



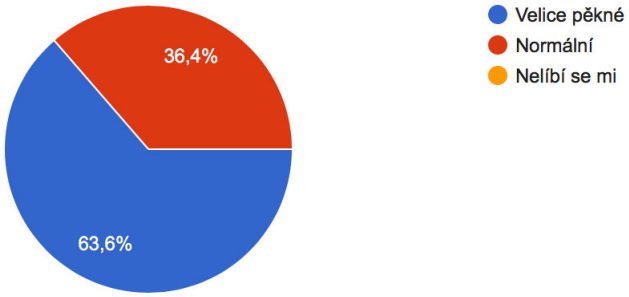
Obrázek B.2: Výsledek dotazníku - Věk

Zkušenosti s hrami typu Tower Defense



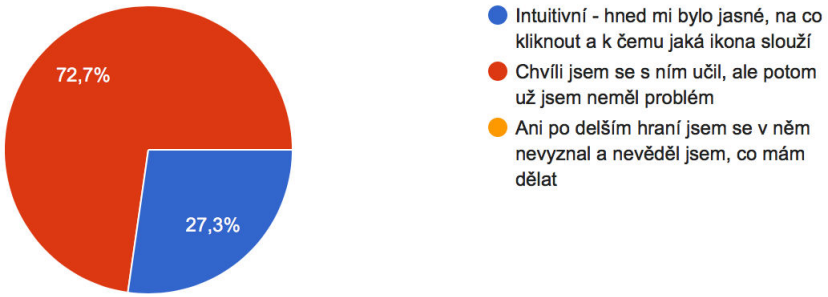
Obrázek B.3: Výsledek dotazníku - Zkušenosti s hrami typu Tower Defense

Jak jste spokojen(a) s grafickým provedením hry



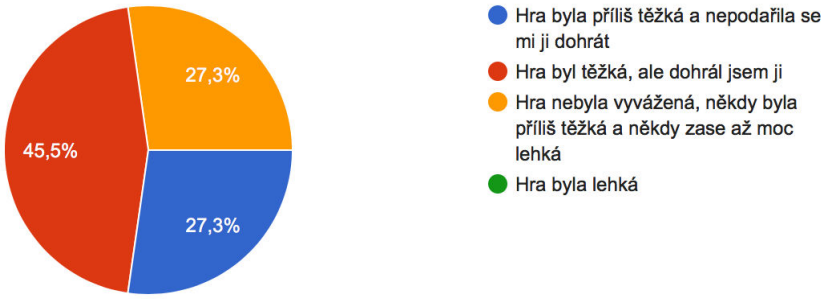
Obrázek B.4: Výsledek dotazníku - Jak jste spokojen(a) s grafickým provedením hry

Jaké bylo uživatelské rozhraní



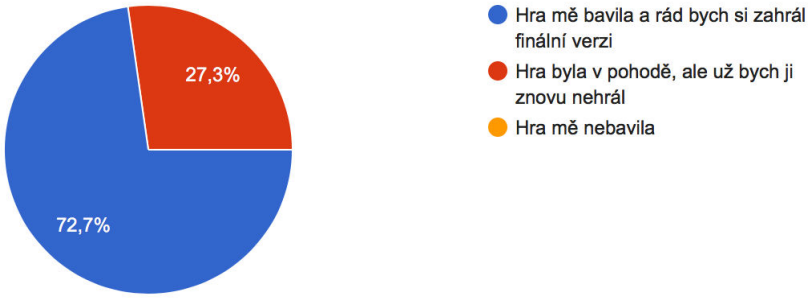
Obrázek B.5: Výsledek dotazníku - Jaké bylo uživatelské rozhraní

Obtížnost hry



Obrázek B.6: Výsledek dotazníku - Obtížnost hry

Bavila Vás hra?



Obrázek B.7: Výsledek dotazníku - Bavila Vás hra?

Připomínky ke hře (problémy, návrhy na zlepšení)

Občas se stalo, že věž střílela úplně jinam než měla. Věže mě přišli zbytečně moc drahé. Měl jsem velký problém s umístěním mapy na bílý stůl nebo u s umístěním při horších světelných podmínkách.
Telefon se mi docela zahříval.
Chybělo mi tam nějaké zobrazení dostřelu věží.
Občas trošku problém s umísťování mapy.
Trošku bych zlevnil věže.
V některých vlnách se hra hrozně seká a celou dobu hraní se mi příšerně hřeje telefon.
Hodilo by se tlačítko na zrychlení celé hry, některé vlny jsou dlouhé a nudné.
Strašně se mi zahříval telefon.
Hra stojí na solidním základu, a koncept, pokud by se rozšířil o několik úrovní, jak na technické, logické a také vizuální, má bezpochyb potenciál
Při hraní ve vlaku mi mapa hry, během hraní "utekla" oken proti směru pohybu vlaku.

Obrázek B.8: Výsledek dotazníku - Připomínky ke hře (problémy, návrhy na zlepšení)